Carlo Finnegan
Student no. 18379666

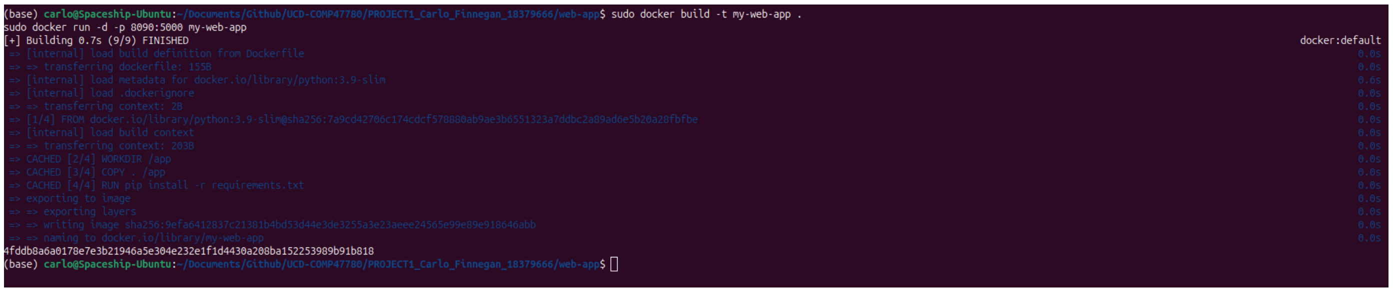
Exercise 1

First navigate to web app directory then execute the following commands.

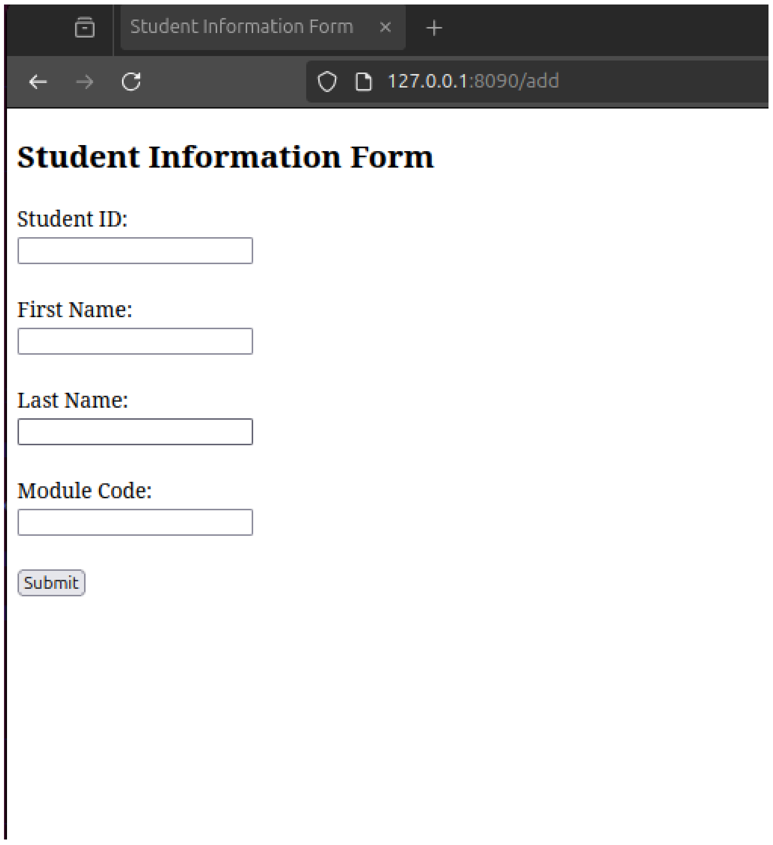
sudo docker build -t my-web-app .

sudo docker run -d -p 8090:5000 my-web-app

I started by dockerizing a simple Flask web application. After setting up the folder structure and creating a Dockerfile, I used the python:3.9-slim base image to run the app in a container. I configured it to be accessible on port 8090 of my host machine, and I was able to navigate to the /add and /all routes. At this point, the form wasn't functional, as it still needed to be connected to the API, which I worked on in the next exercise. Below are screenshots of service working



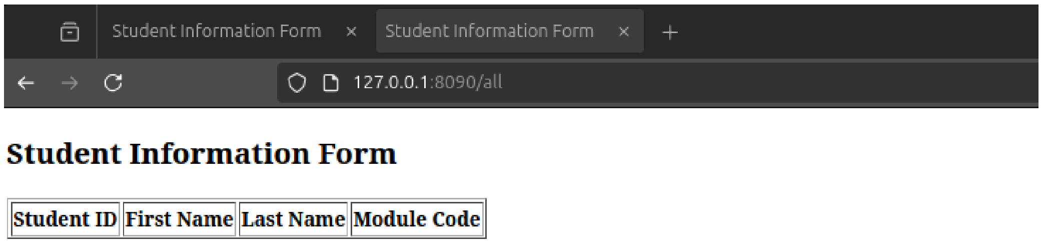
```
(base) carlo@Spaceship-Ubuntu:~/Documents/Github/UCD-COMP47780/PROJECT1_Carlo_Finnegan_18379666/web-app$ sudo docker build -t my-web-app .
sudo docker run -d -p 8090:5000 my-web-app
[+] Building 0.7s (9/9) FINISHED                                                                                                              docker:default
 => [internal] load build definition from Dockerfile                                                                                                    0.0s
 => => transferring dockerfile: 155B                                                                                                                    0.0s
 => [internal] load metadata for docker.io/library/python:3.9-slim                                                                                      0.6s
 => [internal] load .dockerignore                                                                                                                       0.0s
 => => transferring context: 2B                                                                                                                         0.0s
 => [1/4] FROM docker.io/library/python:3.9-slim@sha256:7a9cd42706c174cdcf578880ab9ae3b6551323a7ddbc2a89ad6e5b20a28fbfbe                                0.0s
 => [internal] load build context                                                                                                                       0.0s
 => => transferring context: 203B                                                                                                                       0.0s
 => CACHED [2/4] WORKDIR /app                                                                                                                           0.0s
 => CACHED [3/4] COPY . /app                                                                                                                            0.0s
 => CACHED [4/4] RUN pip install -r requirements.txt                                                                                                    0.0s
 => exporting to image                                                                                                                                  0.0s
 => => exporting layers                                                                                                                                 0.0s
 => => writing image sha256:9efa6412837c21381b4bd53d44e3de3255a3e23aeee24565e99e89e918646abb                                                            0.0s
 => => naming to docker.io/library/my-web-app                                                                                                           0.0s
4fddb8a6a0178e7e3b21946a5e304e232e1f1d4430a208ba152253989b91b818
(base) carlo@Spaceship-Ubuntu:~/Documents/Github/UCD-COMP47780/PROJECT1_Carlo_Finnegan_18379666/web-app$ 
```



Student Information Form

Student ID:	
First Name:	
Last Name:	
Module Code:	

Carlo Finnegan
Student no. 18379666


Student Information Form | Student Information Form
127.0.0.1:8090/all

Student Information Form

Student ID	First Name	Last Name	Module Code

Exercise 2

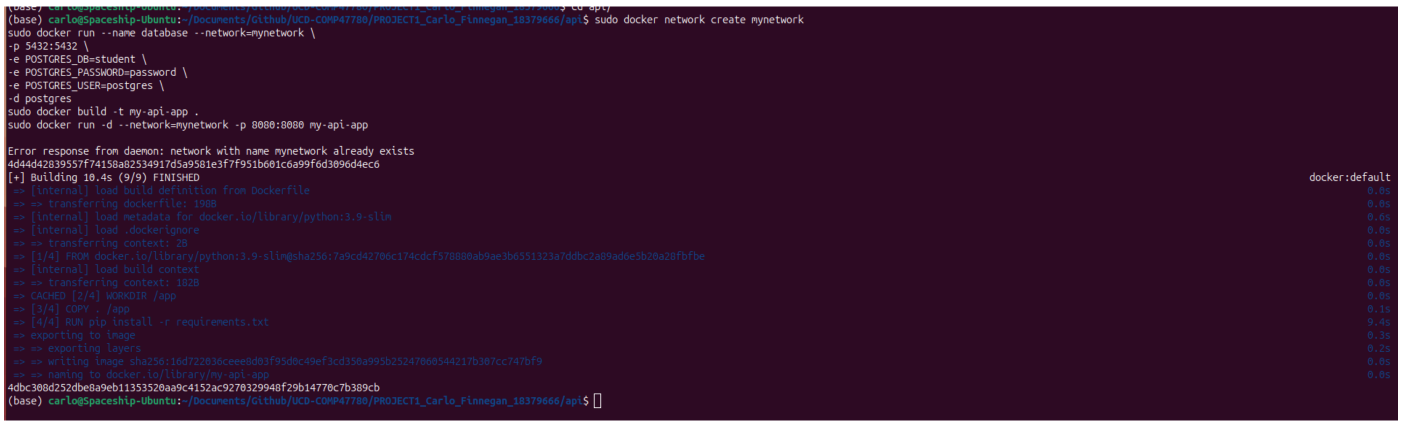
First navigate to api directory then execute the following commands.

sudo docker network create mynetwork
sudo docker run --name database --network=mynetwork \
-p 5432:5432 \
-e POSTGRES_DB=student \
-e POSTGRES_PASSWORD=password \
-e POSTGRES_USER=postgres \
-d postgres

sudo docker build -t my-api-app .
sudo docker run -d --network=mynetwork -p 8080:8080 my-api-app

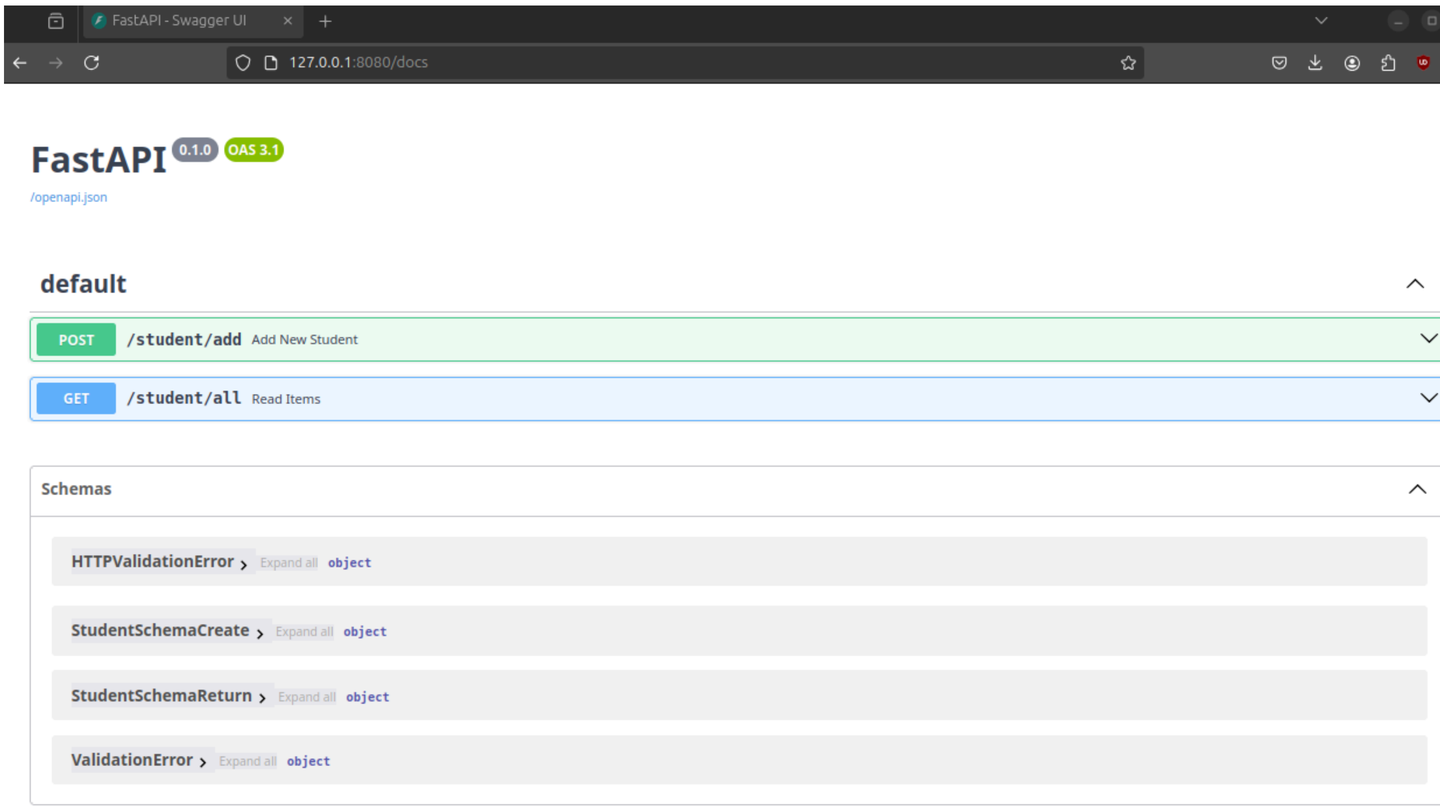
I created an API using FastAPI, again containerizing it with a Dockerfile and the python:3.9-slim base image. The API was set up to run on port 8080. I also set up a temporary PostgreSQL database in a separate container to connect the API to. Once everything was running, I tested the API using FastAPI's interactive docs at 127.0.0.1:8080/docs and verified that I could send and retrieve data. Below are screenshots of api working

Carlo Finnegan
Student no. 18379666



```
(base) carlo@Spaceship-Ubuntu:~/Documents/Github/UCD-COMP47780/PROJECT1_Carlo_Finnegan_18379666/api$ sudo docker network create mynetwork
sudo docker run --name database --network=mynetwork \
-p 5432:5432 \
-e POSTGRES_DB=student \
-e POSTGRES_PASSWORD=password \
-e POSTGRES_USER=postgres \
-d postgres
sudo docker build -t my-api-app .
sudo docker run -d --network=mynetwork -p 8080:8080 my-api-app

Error response from daemon: network with name mynetwork already exists
4d44d42839557f74158a82534917d5a9581e3f7f951b601c6a99f6d3096d4ec6
[+] Building 10.4s (9/9) FINISHED                                                          docker:default
 => [internal] load build definition from Dockerfile                                                 0.0s
 => => transferring dockerfile: 198B                                                                 0.0s
 => [internal] load metadata for docker.io/library/python:3.9-slim                                   0.6s
 => [internal] load .dockerignore                                                                    0.0s
 => => transferring context: 2B                                                                      0.0s
 => [1/4] FROM docker.io/library/python:3.9-slim@sha256:7a9cd42706c174cdcf578880ab9ae3b6551323a7ddbc2a89ad6e5b20a28fbfbe  0.0s
 => [internal] load build context                                                                    0.0s
 => => transferring context: 182B                                                                    0.0s
 => CACHED [2/4] WORKDIR /app                                                                        0.0s
 => [3/4] COPY . /app                                                                                0.1s
 => [4/4] RUN pip install -r requirements.txt                                                        9.4s
 => exporting to image                                                                               0.3s
 => => exporting layers                                                                              0.2s
 => => writing image sha256:16d722036ceee8d03f95d0c49ef3cd350a995b25247060544217b307cc747bf9         0.0s
 => => naming to docker.io/library/my-api-app                                                        0.0s
4dbc308d252dbe8a9eb11353520aa9c4152ac9270329948f29b14770c7b389cb
(base) carlo@Spaceship-Ubuntu:~/Documents/Github/UCD-COMP47780/PROJECT1_Carlo_Finnegan_18379666/api$
```



FastAPI - Swagger UI
127.0.0.1:8080/docs

FastAPI 0.1.0 OAS 3.1
/openapi.json

default

POST /student/add Add New Student

GET /student/all Read Items

Schemas

HTTPValidationError > Expand all object

StudentSchemaCreate > Expand all object

StudentSchemaReturn > Expand all object

ValidationError > Expand all object

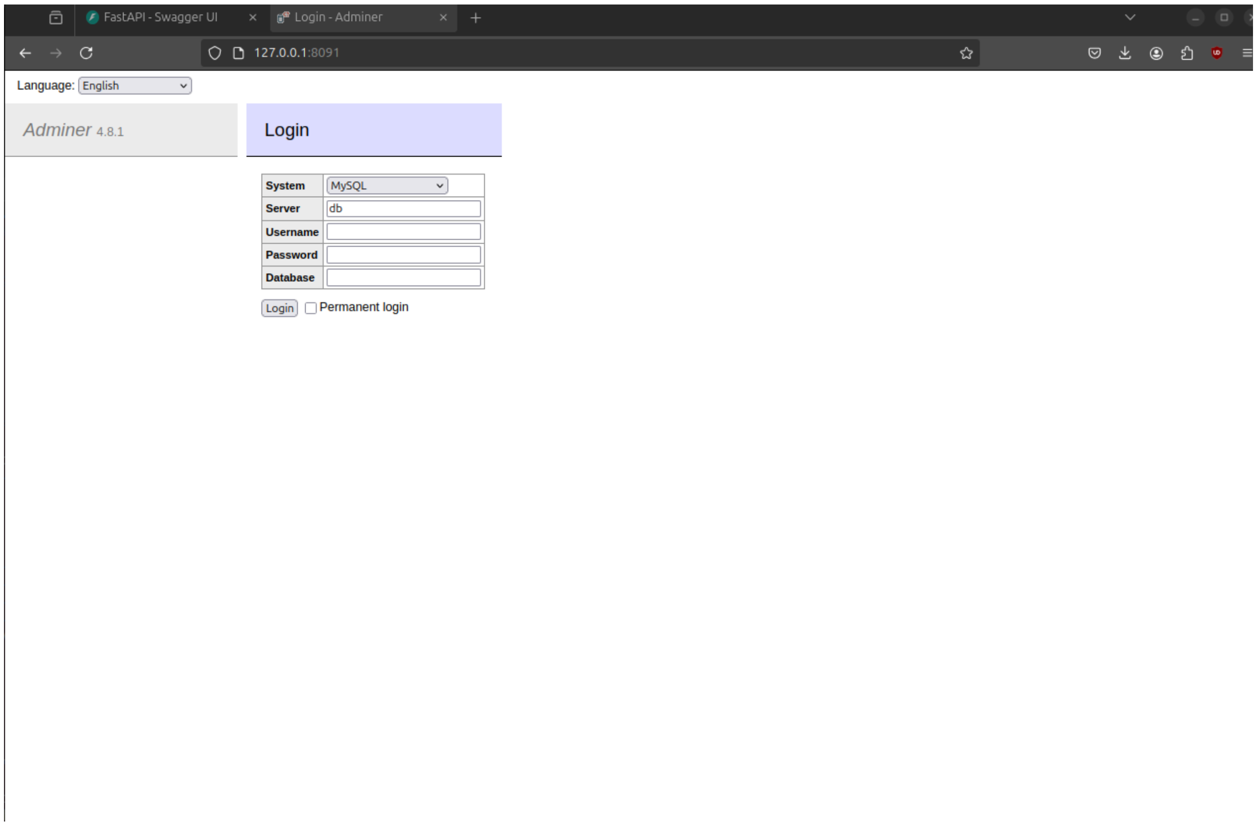
Above is me connected to the FastAPI

Exercise 3

In this part of the project, I set up a PostgreSQL database and used Adminer to visualize and manage the database, demonstrating how to use different containers with Docker. I ran the following commands to setup the database and connect it to adminer

Carlo Finnegan
Student no. 18379666

sudo docker network create backend
sudo docker run --name database --network backend -e POSTGRES_PASSWORD=password -d postgres
sudo docker run -d --network backend -p 8091:8080 adminer

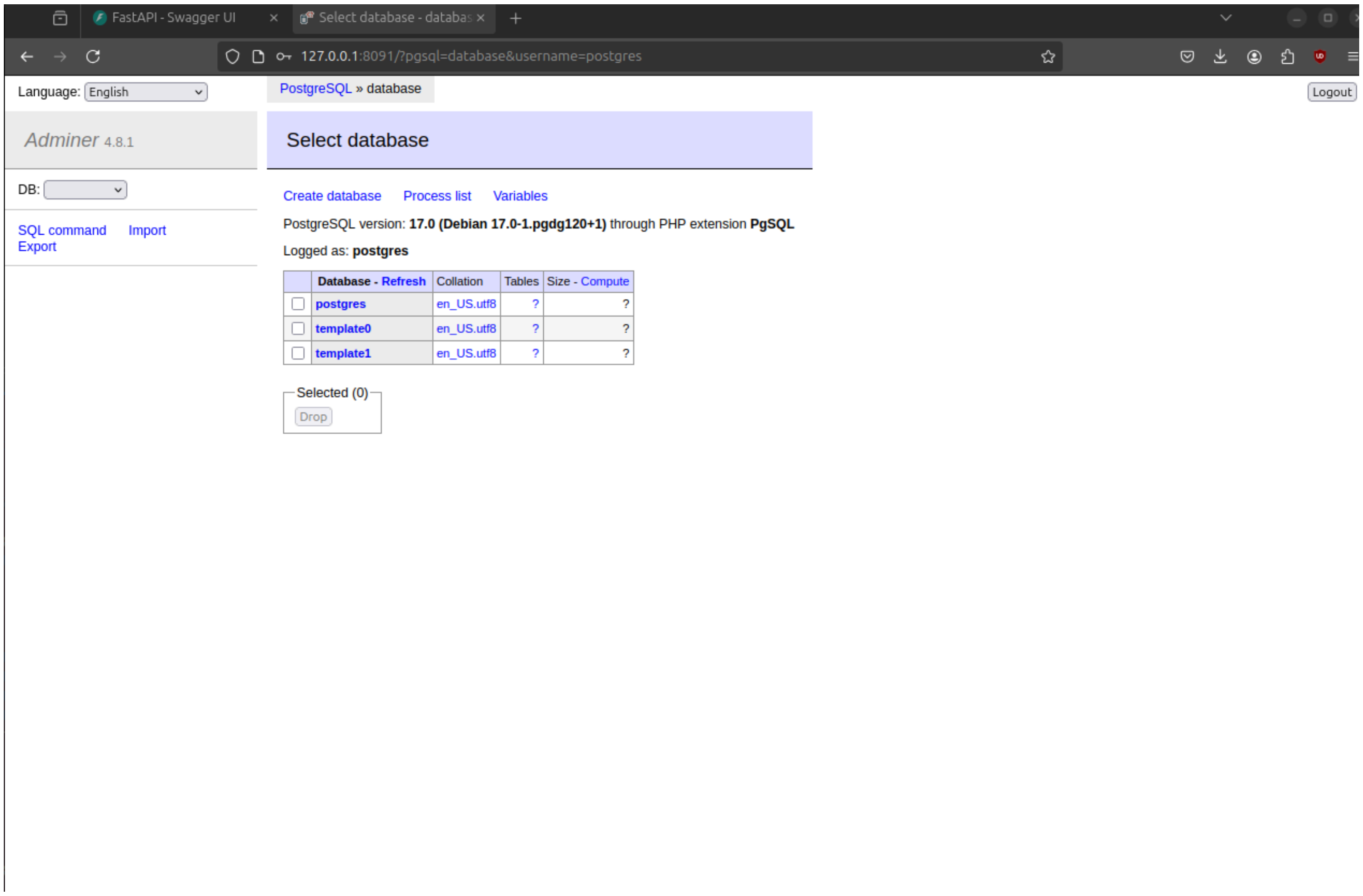


System	MySQL
Server	db
Username	
Password	
Database	

I logged into the database using the following parameters and making sure to set the database system to postgres.

Server: database
Username: postgres
Password: password

Carlo Finnegan
Student no. 18379666



PostgreSQL » database

Select database

Create database Process list Variables

PostgreSQL version: **17.0 (Debian 17.0-1.pgdg120+1)** through PHP extension **PgSQL**

Logged as: **postgres**

	Database - Refresh	Collation	Tables	Size - Compute
☐	postgres	en_US.utf8	?	?
☐	template0	en_US.utf8	?	?
☐	template1	en_US.utf8	?	?

Selected (0)
Drop

Here you can see me connected to the database

Exercise 4

In this step, we first set up a persistent volume for the database and define environment variables for the database credentials in a .env file. Next, I configured the adminer service to depend on the database and attach both services to the backend network. Then I configure the web-app to depend on both the database and api services by attaching it to the frontend network and also mapping it to port 8090 on the host.I run the code sudo docker compose up to run my compose file and start all the containers. The configuration can be seen in the individual files. Which is covered in the video

```
sudo docker compose up
```

The associated file can be seen working in the video in zipped folder:
The service can be seen working in the video alter the database by using the form and it being relayed also altering it directly accessing the database using adminer. I changed the associated files in this step to the updated ones.

Carlo Finnegan
Student no. 18379666